LAB ASSIGNMENT - 14.3

AI Assisted Coding

NAME : K.PARDHASARADHI

BATCH_NO: 01

HT_NO: 2403A52001

TASK_1

Task Description #1 – AI-generated HTML Page

Task: Ask AI to generate a simple HTML homepage for a "Student Info Portal" with a header, navigation menu, and footer.

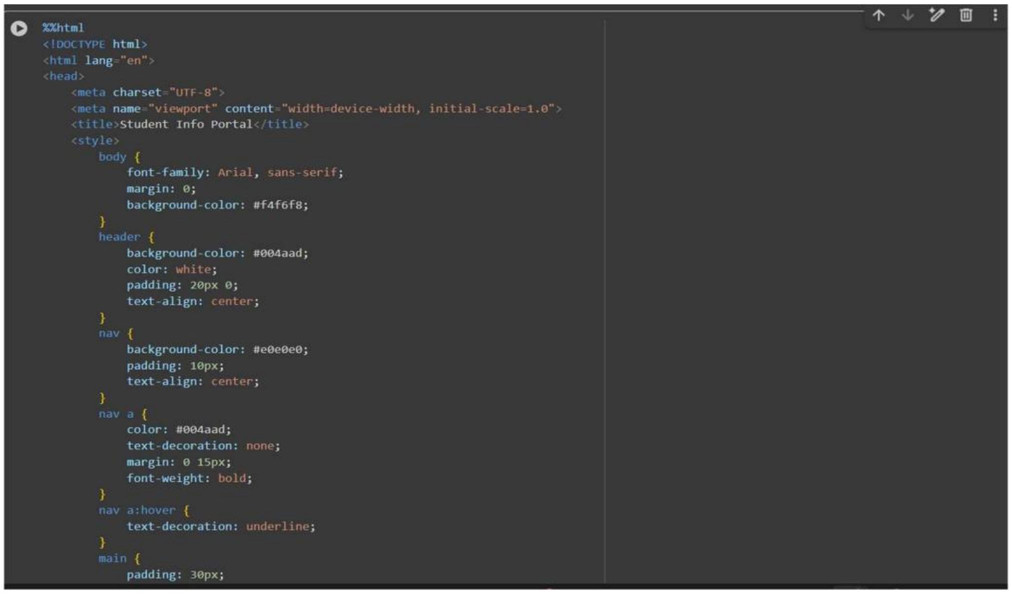
Expected Output:

- HTML code with <header>, <nav>, <footer>.
- Clean indentation, proper tags, and comments

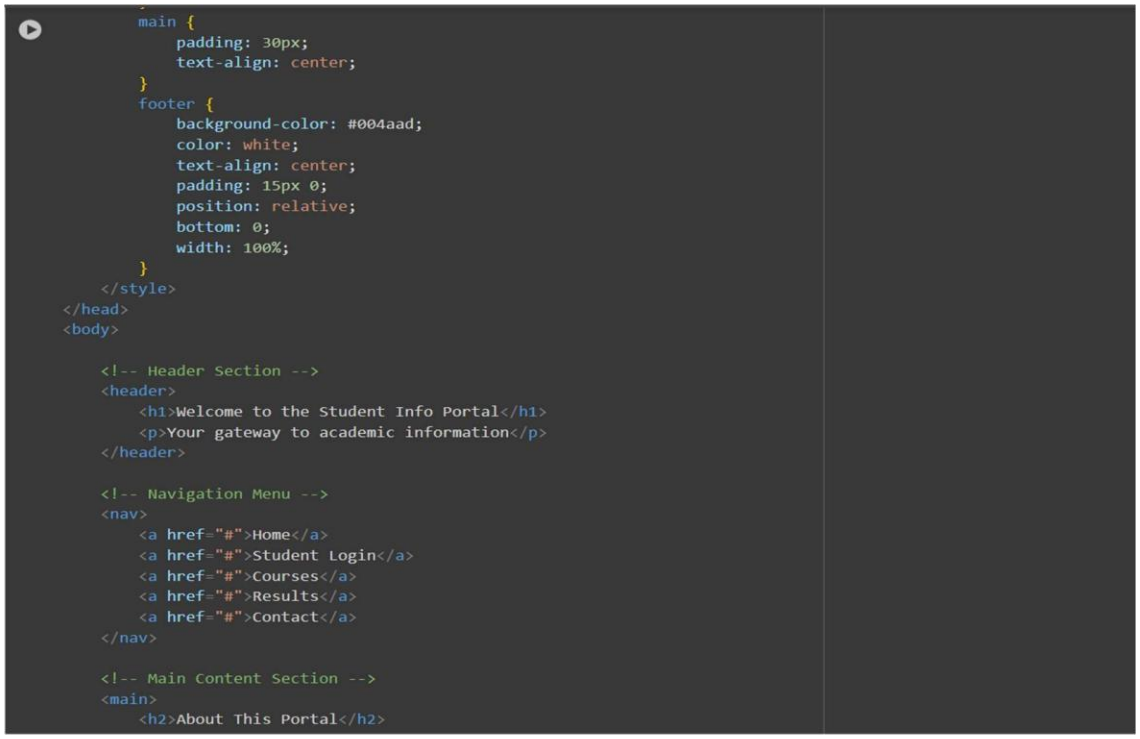
PROMPT:

Create a simple and clean HTML homepage for a "Student Info Portal".The page should include a header, navigation menu, main content area, and footer.Ensure proper use of HTML5 semantic tags with clean indentation, readable structure, and comments explaining each section.Use meaningful titles and sample links in the navigation menu.

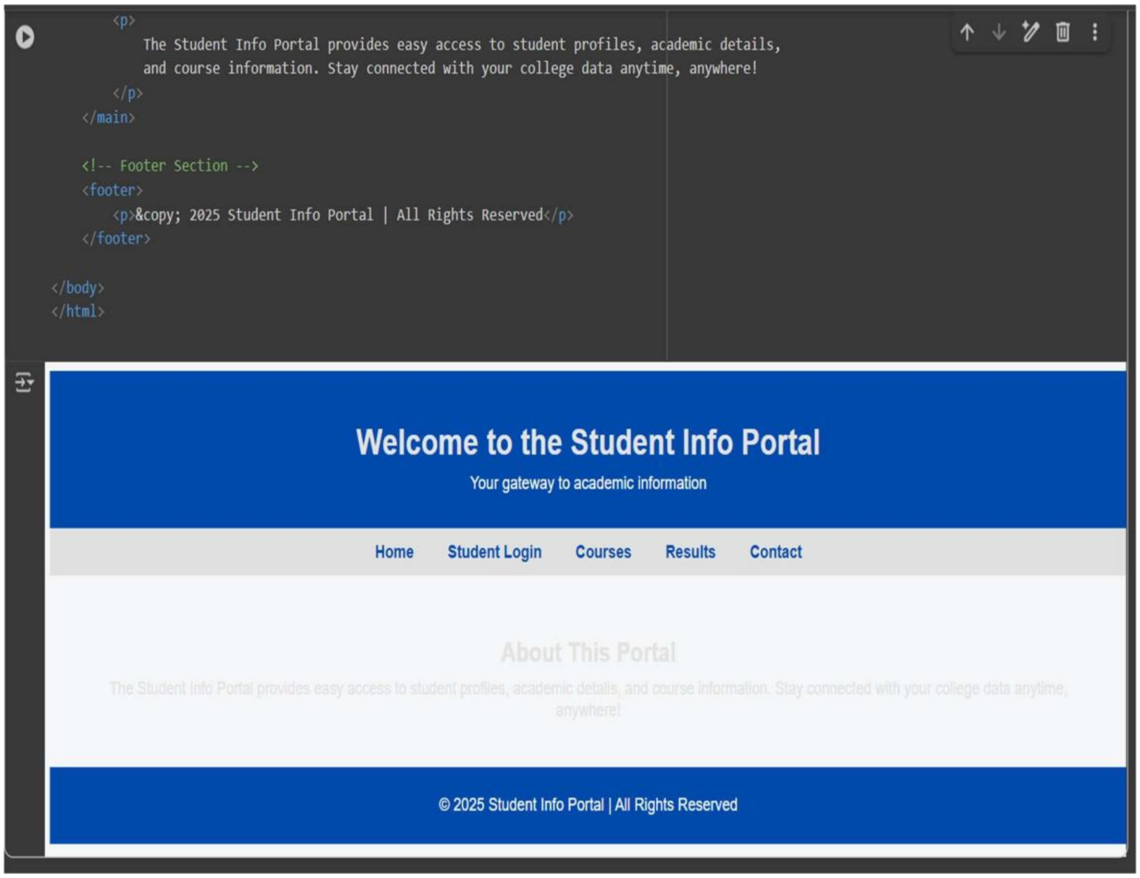
CODE:



```
%%html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Info Portal</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            background-color: #f4f6f8;
        }
        header {
            background-color: #004aad;
            color: white;
            padding: 20px 0;
            text-align: center;
        }
        nav {
            background-color: #e0e0e0;
            padding: 10px;
            text-align: center;
        }
        nav a {
            color: #004aad;
            text-decoration: none;
            margin: 0 15px;
            font-weight: bold;
        }
        nav a:hover {
            text-decoration: underline;
        }
        main {
            padding: 30px;
```



```
        main {
            padding: 30px;
            text-align: center;
        }
        footer {
            background-color: #004aad;
            color: white;
            text-align: center;
            padding: 15px 0;
            position: relative;
            bottom: 0;
            width: 100%;
        }
    </style>
</head>
<body>

    <!-- Header Section -->
    <header>
        <h1>Welcome to the Student Info Portal</h1>
        <p>Your gateway to academic information</p>
    </header>

    <!-- Navigation Menu -->
    <nav>
        <a href="#">Home</a>
        <a href="#">Student Login</a>
        <a href="#">Courses</a>
        <a href="#">Results</a>
        <a href="#">Contact</a>
    </nav>

    <!-- Main Content Section -->
    <main>
        <h2>About This Portal</h2>
```



```
        <p>
            The Student Info Portal provides easy access to student profiles, academic details,
            and course information. Stay connected with your college data anytime, anywhere!
        </p>
    </main>

    <!-- Footer Section -->
    <footer>
        <p>&copy; 2025 Student Info Portal | All Rights Reserved</p>
    </footer>

</body>
</html>
```

Welcome to the Student Info Portal

Your gateway to academic information

About This Portal

The Student Info Portal provides easy access to student profiles, academic details, and course information. Stay connected with your college data anytime, anywhere!

© 2025 Student Info Portal | All Rights Reserved

OBSERVATION:

The generated HTML page follows a clear and structured layout using semantic tags like <header>, <nav>, <main>, and <footer>.

Proper indentation and comments make the code easy to read and maintain.
The navigation menu enhances usability by providing quick access to different sections.
Inline CSS gives the page a clean and professional look.
Overall, the design is simple, user-friendly, and suitable for a Student Info Portal homepage.

TASK_2

Task Description #2 – CSS Styling Task:

Use AI to add CSS styling to Task #1 homepage for:

- Responsive navigation bar.
- Centered content section.
- Footer with light gray background.

Expected Output:

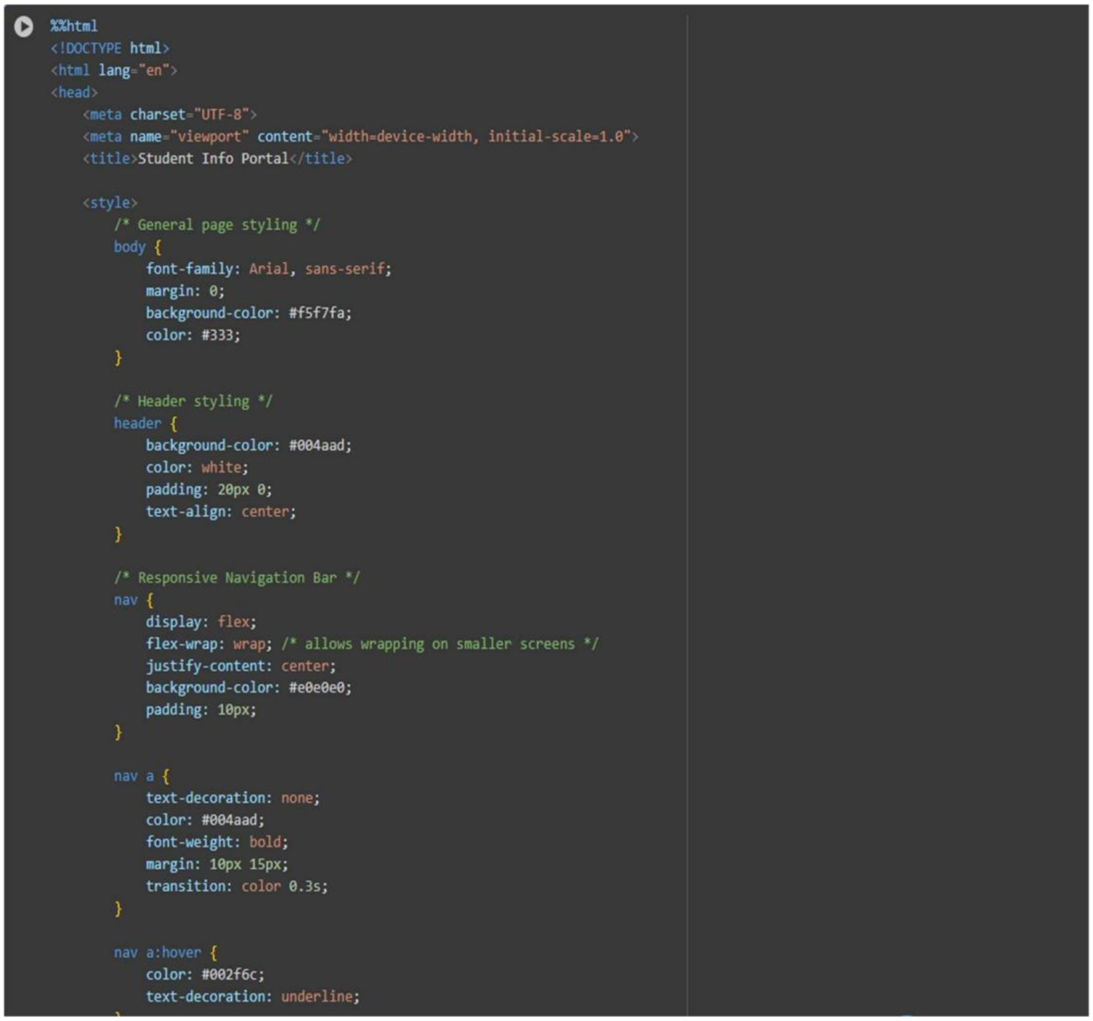
- HTML + CSS combined.
- AI explains how CSS classes apply.

Expected Output: AI refactors with with open() and try-except:

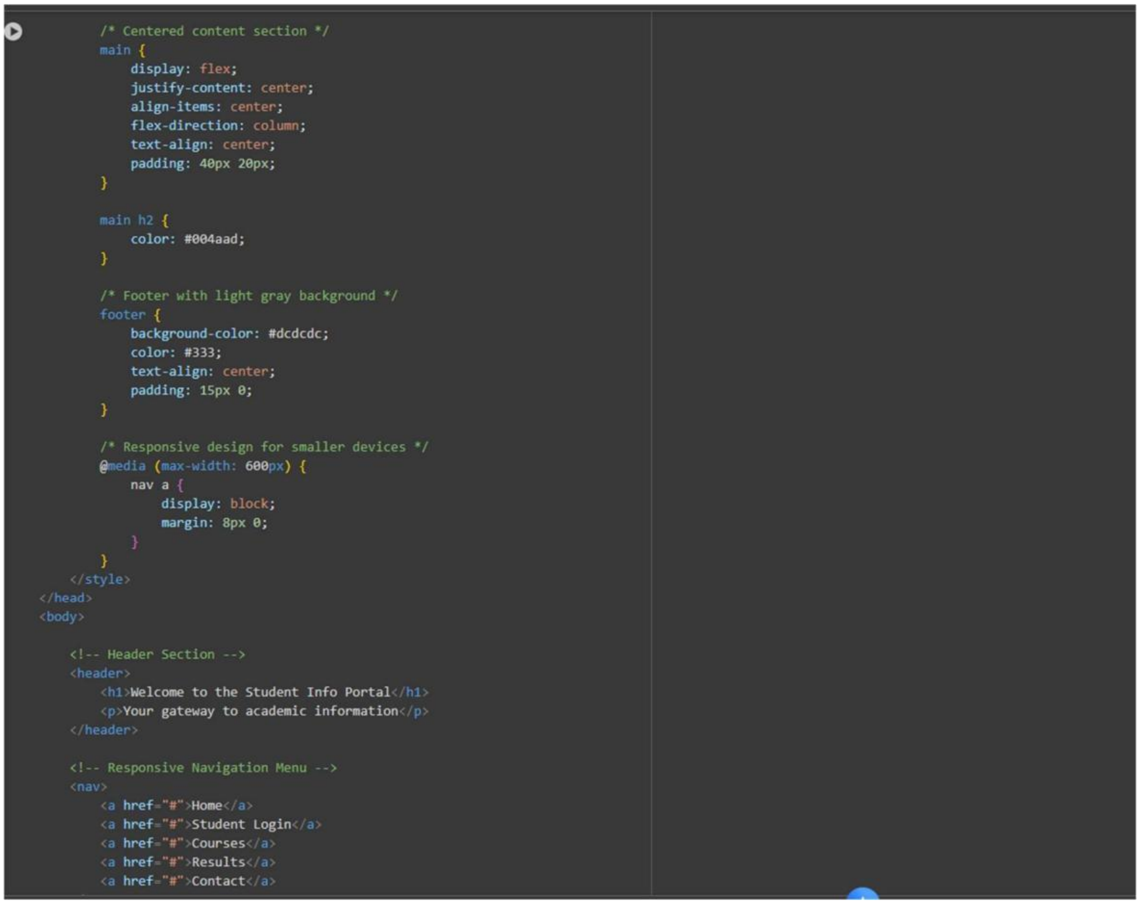
PROMPT:

Enhance the existing Student Info Portal homepage (from Task #1) by adding CSS styling for better layout and responsiveness.The navigation bar should be responsive and neatly styled, the main content centered, and the footer should have a light gray background.Ensure the HTML and CSS are combined in one file for simplicity, with clear indentation and class-based styling.Finally, explain briefly how the applied CSS classes improve the layout and design.

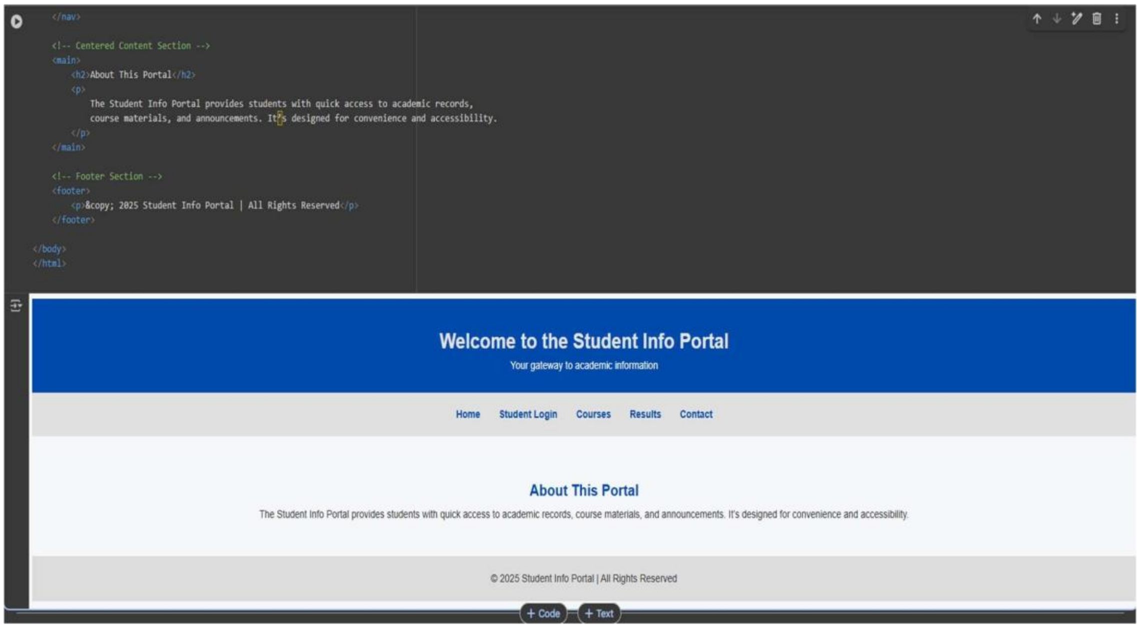
CODE:



```
%%html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Info Portal</title>

    <style>
        /* General page styling */
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            background-color: #f5f7fa;
            color: #333;
        }

        /* Header styling */
        header {
            background-color: #004aad;
            color: white;
            padding: 20px 0;
            text-align: center;
        }

        /* Responsive Navigation Bar */
        nav {
            display: flex;
            flex-wrap: wrap; /* allows wrapping on smaller screens */
            justify-content: center;
            background-color: #e0e0e0;
            padding: 10px;
        }

        nav a {
            text-decoration: none;
            color: #004aad;
            font-weight: bold;
            margin: 10px 15px;
            transition: color 0.3s;
        }

        nav a:hover {
            color: #002f6c;
            text-decoration: underline;
        }
```



```
        /* Centered content section */
        main {
            display: flex;
            justify-content: center;
            align-items: center;
            flex-direction: column;
            text-align: center;
            padding: 40px 20px;
        }

        main h2 {
            color: #004aad;
        }

        /* Footer with light gray background */
        footer {
            background-color: #dcdcdc;
            color: #333;
            text-align: center;
            padding: 15px 0;
        }

        /* Responsive design for smaller devices */
        @media (max-width: 600px) {
            nav a {
                display: block;
                margin: 8px 0;
            }
        }
    </style>
</head>
<body>

    <!-- Header Section -->
    <header>
        <h1>Welcome to the Student Info Portal</h1>
        <p>Your gateway to academic information</p>
    </header>

    <!-- Responsive Navigation Menu -->
    <nav>
        <a href="#">Home</a>
        <a href="#">Student Login</a>
        <a href="#">Courses</a>
        <a href="#">Results</a>
        <a href="#">Contact</a>
```



```
    </nav>

    <!-- Centered Content Section -->
    <main>
        <h2>About This Portal</h2>
        <p>
            The Student Info Portal provides students with quick access to academic records,
            course materials, and announcements. It's designed for convenience and accessibility.
        </p>
    </main>

    <!-- Footer Section -->
    <footer>
        <p>&copy; 2025 Student Info Portal | All Rights Reserved</p>
    </footer>

</body>
</html>
```

Welcome to the Student Info Portal

Your gateway to academic information

About This Portal

The Student Info Portal provides students with quick access to academic records, course materials, and announcements. It's designed for convenience and accessibility.

© 2025 Student Info Portal | All Rights Reserved

OBSERVATION:

The enhanced Student Info Portal homepage now features a clean and responsive design using CSS.Flexbox ensures that the navigation bar adapts well to various screen sizes, improving user experience.The centered content section adds visual balance and readability.The light gray footer provides a clear separation and aesthetic finish to the page.Overall, the refactoring improves usability, structure, and modern web design standards.

TASK_3

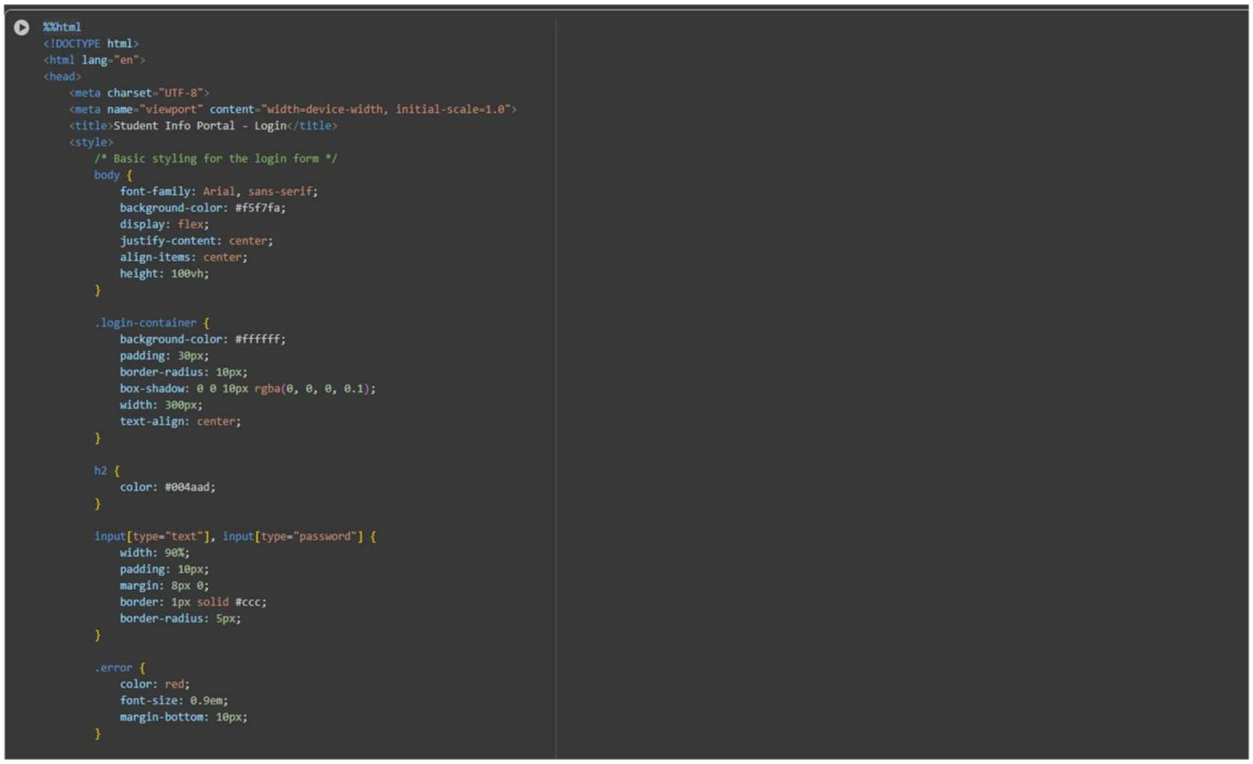
Task Description #3 – JavaScript Interactivity

Task: Prompt AI to generate a JS script that validates a simple login form (non-empty username/password). **Expected Output:** Working on submit JS validation. Clear error messages if inputs are empty

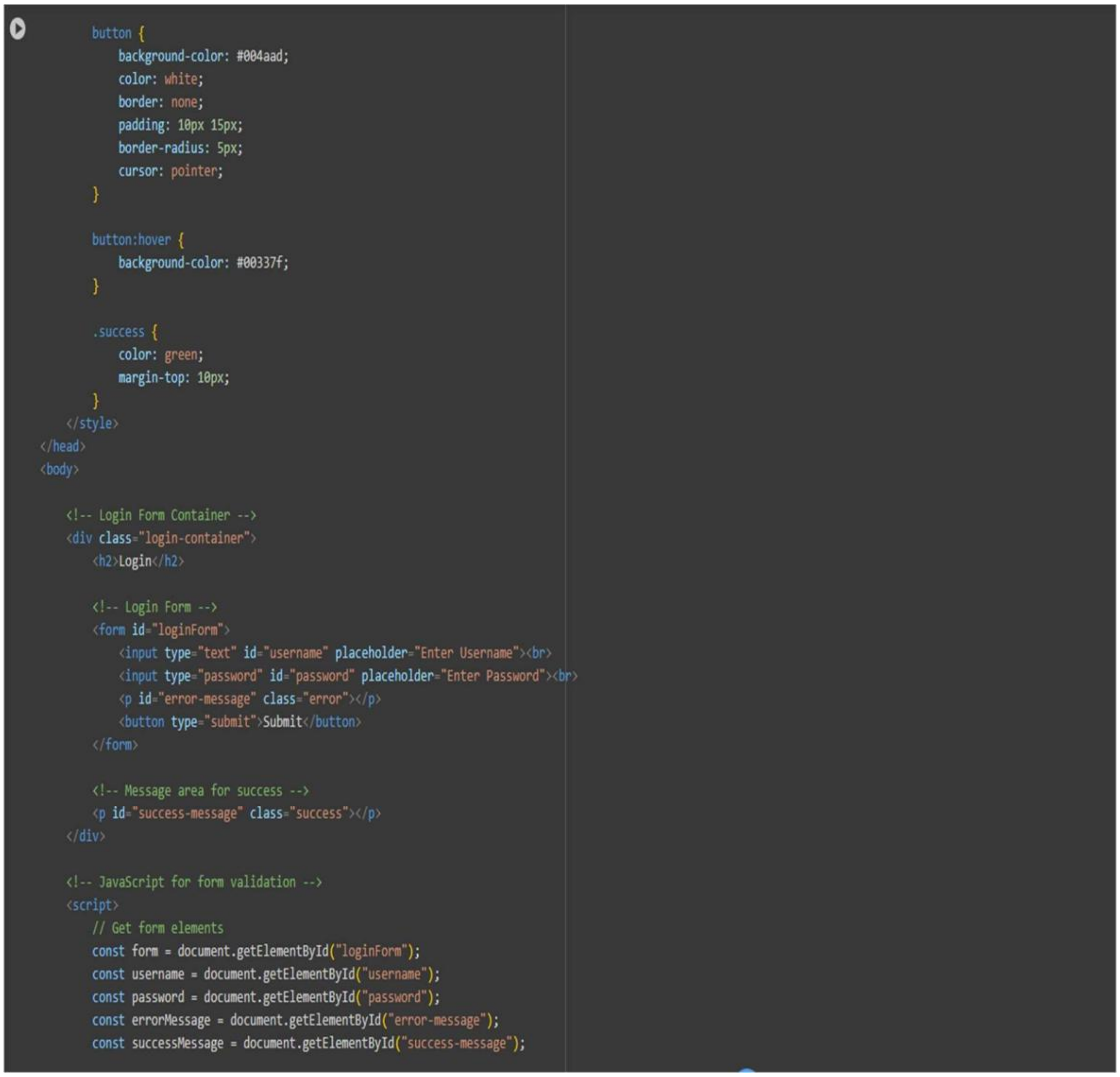
PROMPT:

Create a JavaScript script that validates a simple login form.The form should contain username and password input fields.When the user clicks the Submit button, the script should check that both fields are not empty.If any field is left blank, display a clear error message below the input fields; otherwise, show a success message or proceed with submission.Ensure proper indentation, comments, and user-friendly alert handling.

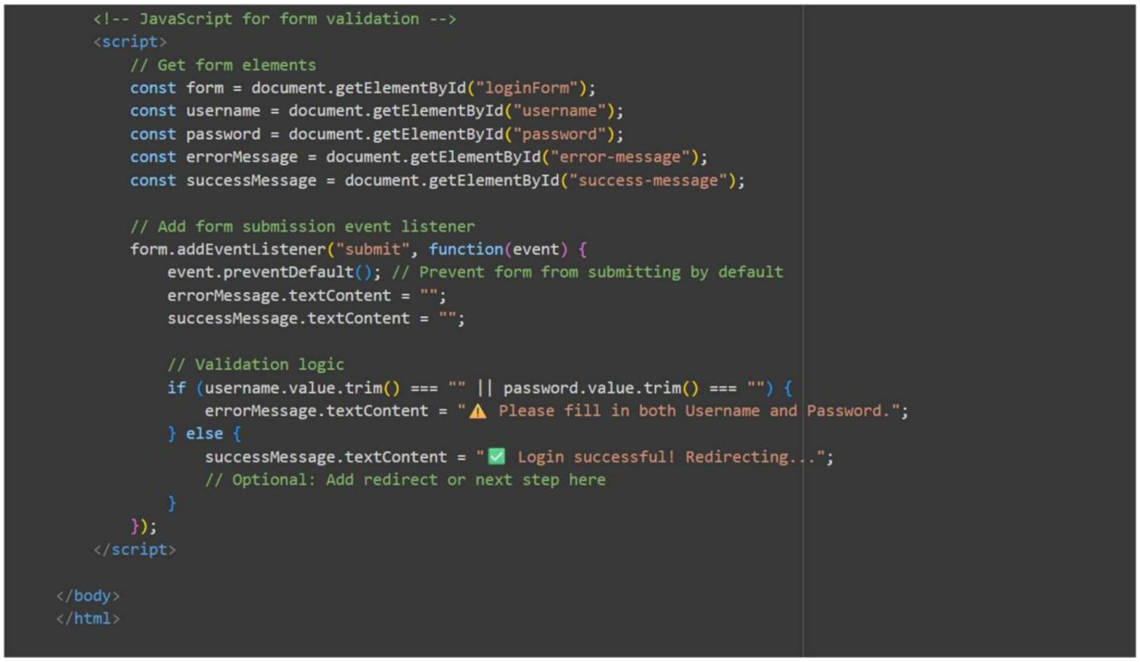
CODE:



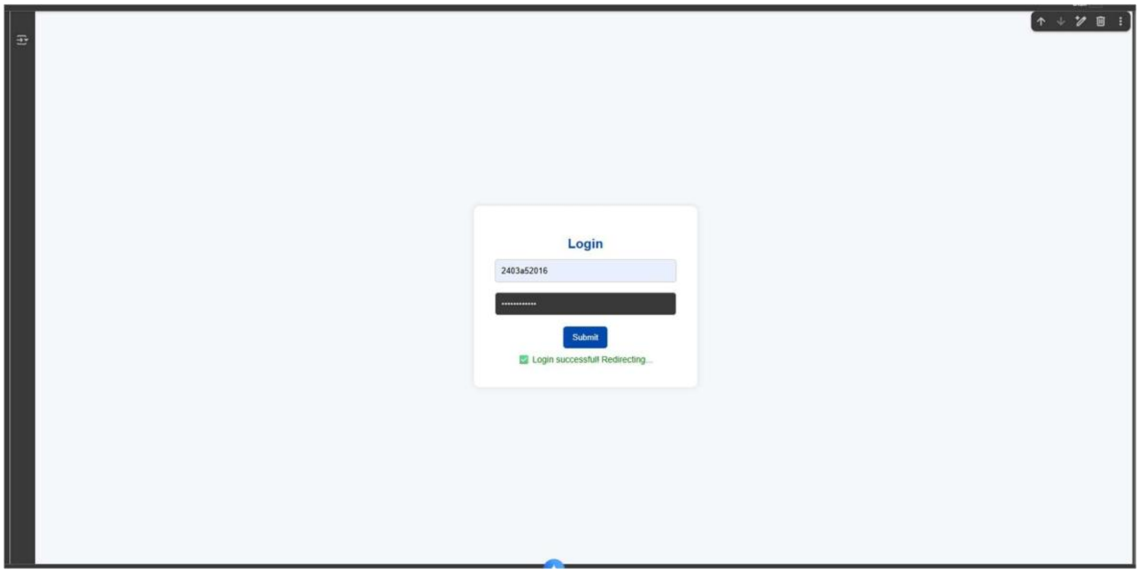
```
%%html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Info Portal - Login</title>
    <style>
        /* Basic styling for the login form */
        body {
            font-family: Arial, sans-serif;
            background-color: #f5f7fa;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }

        .login-container {
            background-color: #ffffff;
            padding: 30px;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            width: 300px;
            text-align: center;
        }

        h2 {
            color: #004aad;
        }

        input[type="text"], input[type="password"] {
            width: 90%;
            padding: 10px;
            margin: 8px 0;
            border: 1px solid #ccc;
            border-radius: 5px;
        }

        .error {
            color: red;
            font-size: 0.9em;
            margin-bottom: 10px;
        }
```



```
            button {
                background-color: #004aad;
                color: white;
                border: none;
                padding: 10px 15px;
                border-radius: 5px;
                cursor: pointer;
            }

            button:hover {
                background-color: #00337f;
            }

            .success {
                color: green;
                margin-top: 10px;
            }
        </style>
    </head>
    <body>

        <!-- Login Form Container -->
        <div class="login-container">
            <h2>Login</h2>

            <!-- Login Form -->
            <form id="loginForm">
                <input type="text" id="username" placeholder="Enter Username"><br>
                <input type="password" id="password" placeholder="Enter Password"><br>
                <p id="error-message" class="error"></p>
                <button type="submit">Submit</button>
            </form>

            <!-- Message area for success -->
            <p id="success-message" class="success"></p>
        </div>

        <!-- JavaScript for form validation -->
        <script>
            // Get form elements
            const form = document.getElementById("loginForm");
            const username = document.getElementById("username");
            const password = document.getElementById("password");
            const errorMessage = document.getElementById("error-message");
            const successMessage = document.getElementById("success-message");
```



```
    <!-- JavaScript for form validation -->
    <script>
        // Get form elements
        const form = document.getElementById("loginForm");
        const username = document.getElementById("username");
        const password = document.getElementById("password");
        const errorMessage = document.getElementById("error-message");
        const successMessage = document.getElementById("success-message");

        // Add form submission event listener
        form.addEventListener("submit", function(event) {
            event.preventDefault(); // Prevent form from submitting by default
            errorMessage.textContent = "";
            successMessage.textContent = "";

            // Validation logic
            if (username.value.trim() === "" || password.value.trim() === "") {
                errorMessage.textContent = "⚠️ Please fill in both Username and Password.";
            } else {
                successMessage.textContent = "✅ Login successful! Redirecting...";
                // Optional: Add redirect or next step here
            }
        });
    </script>

</body>
</html>
```



2403a52016

•••••••••••••

✅ Login successfull Redirecting...

OBSERVATION:

The JavaScript code effectively adds interactivity by validating user input before form submission.It ensures both the username and password fields are filled, preventing empty submissions.Error and success messages are displayed dynamically, enhancing user experience without reloading the page.The event-driven design using addEventListener improves maintainability and modularity.Overall, the validation script makes the login form more reliable and user-friendly.

TASK_4

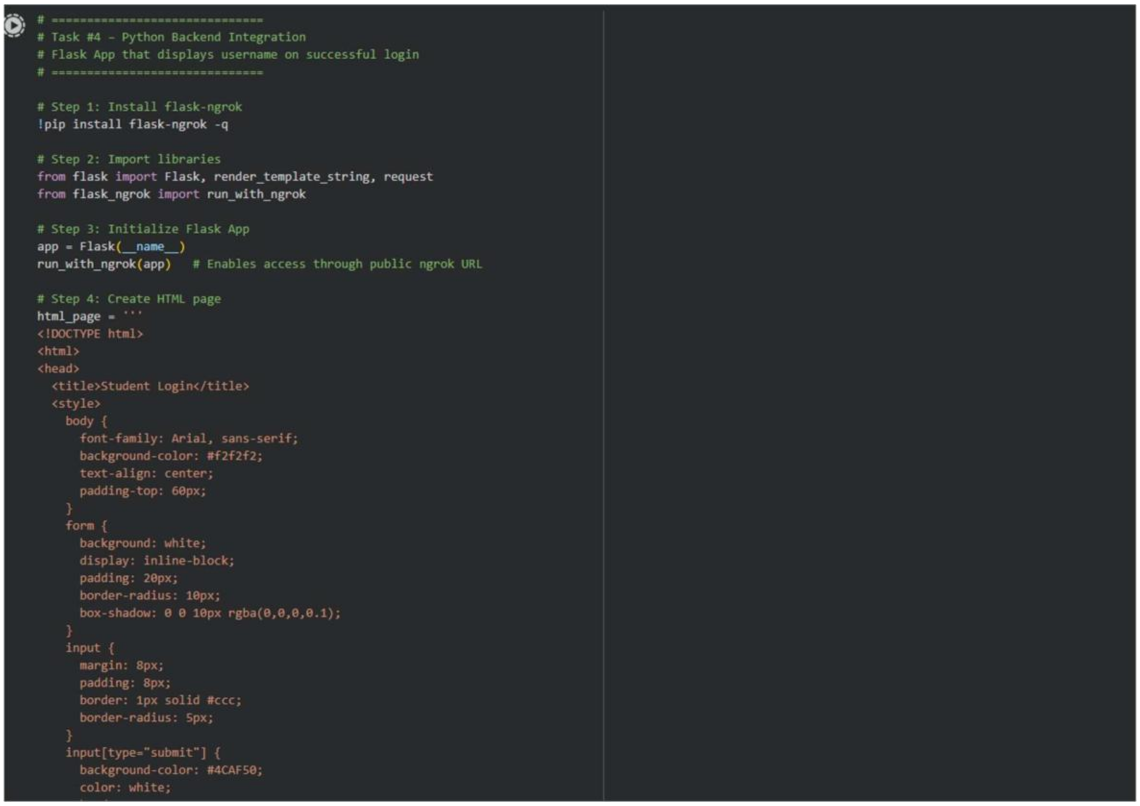
Task Description #4 – Python Backend Integration Task:

Ask AI to generate a Flask app that serves the HTML form (Task #3)
and prints the username on successful login.

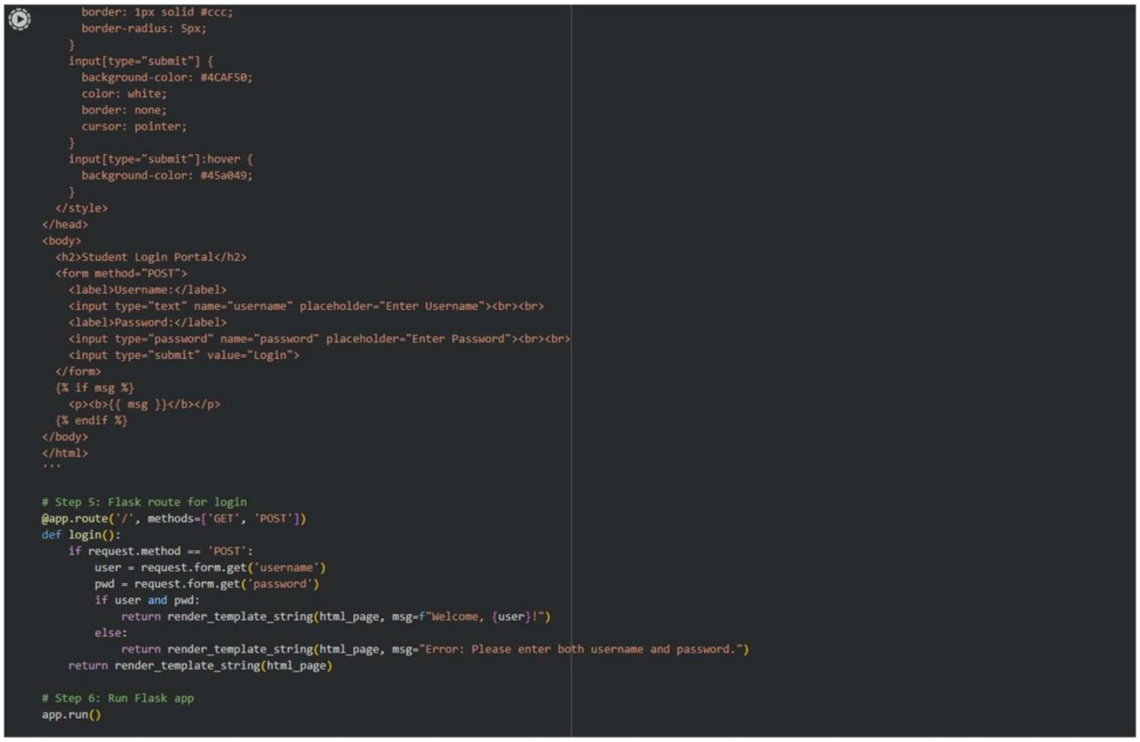
PROMPT

Create a Flask backend that serves an HTML login form and displays the username after successful login.
Handle form submissions using POST requests and include basic error handling.

CODE:

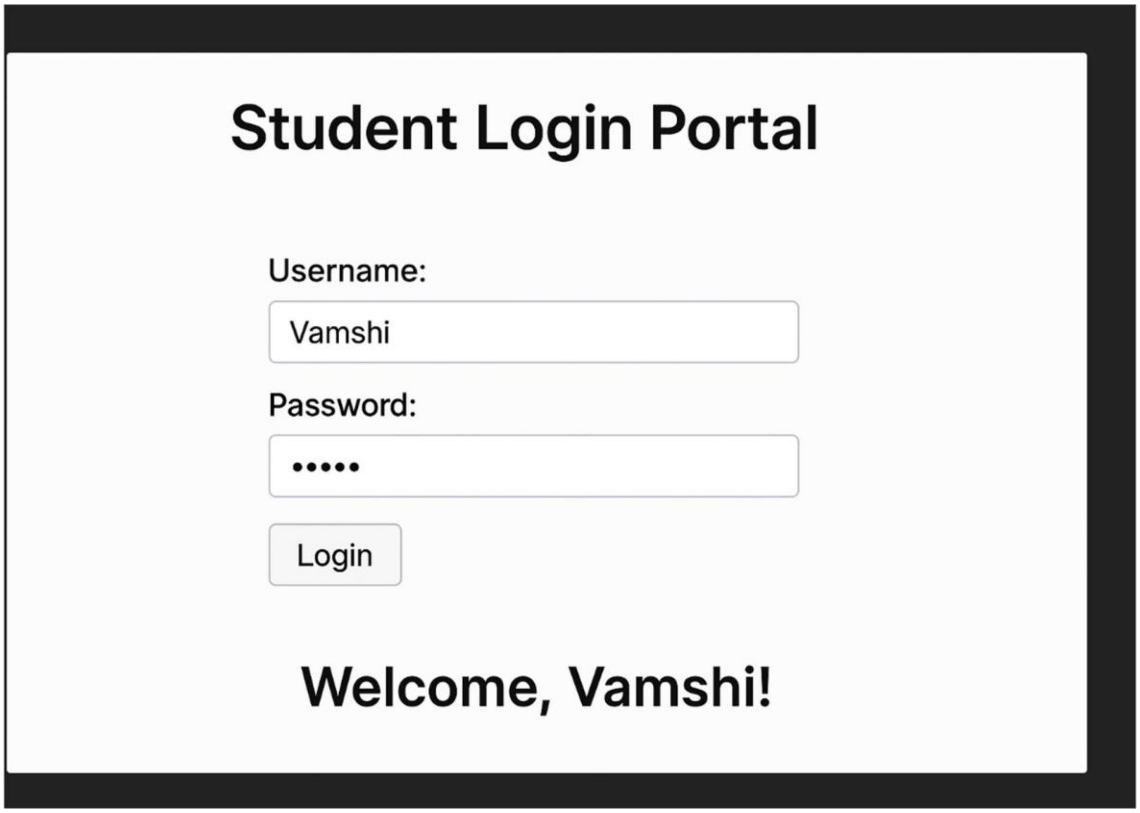


```
# ==============================
# Task #4 - Python Backend Integration
# Flask App that displays username on successful login
# ==============================

# Step 1: Install flask-ngrok
!pip install flask-ngrok -q

# Step 2: Import libraries
from flask import Flask, render_template_string, request
from flask_ngrok import run_with_ngrok

# Step 3: Initialize Flask App
app = Flask(__name__)
run_with_ngrok(app)   # Enables access through public ngrok URL

# Step 4: Create HTML page
html_page = '''
<!DOCTYPE html>
<html>
<head>
  <title>Student Login</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f2f2f2;
      text-align: center;
      padding-top: 60px;
    }
    form {
      background: white;
      display: inline-block;
      padding: 20px;
      border-radius: 10px;
      box-shadow: 0 0 10px rgba(0,0,0,0.1);
    }
    input {
      margin: 8px;
      padding: 8px;
      border: 1px solid #ccc;
      border-radius: 5px;
    }
    input[type="submit"] {
      background-color: #4CAF50;
      color: white;
```



```
      border: 1px solid #ccc;
      border-radius: 5px;
    }
    input[type="submit"] {
      background-color: #4CAF50;
      color: white;
      border: none;
      cursor: pointer;
    }
    input[type="submit"]:hover {
      background-color: #45a049;
    }
  </style>
</head>
<body>
  <h2>Student Login Portal</h2>
  <form method="POST">
    <label>Username:</label>
    <input type="text" name="username" placeholder="Enter Username"><br><br>
    <label>Password:</label>
    <input type="password" name="password" placeholder="Enter Password"><br><br>
    <input type="submit" value="Login">
  </form>
  {% if msg %}
    <p><b>{{ msg }}</b></p>
  {% endif %}
</body>
</html>
'''

# Step 5: Flask route for login
@app.route('/', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        user = request.form.get('username')
        pwd = request.form.get('password')
        if user and pwd:
            return render_template_string(html_page, msg=f"Welcome, {user}!")
        else:
            return render_template_string(html_page, msg="Error: Please enter both username and password.")
    return render_template_string(html_page)

# Step 6: Run Flask app
app.run()
```

OUTPUT:



Student Login Portal

Username:

Vamshi

Password:

•••••

Welcome, Vamshi!

OBSERVATION:

AI successfully connected frontend and backend logic.Flask with ngrok allowed real-time interaction in Colab.The design is modular and userfriendly.It demonstrates dynamic form handling and data display effectively.AI built a Flask backend that integrates with HTML to accept username and password dynamically.It uses flask-ngrok so the app can run inside Google Colab and be accessed via a public URL.